Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
Electronic Notes in Theoretical Computer Science 162 (2006) 217–220
www.elsevier.com/locate/entcs

Process Algebras in the Age of Ubiquitous Computing

Kohei Honda[1]

Department of Computing
Imperial College London
London, England

Abstract

We discuss why process theories are such an enchanting field, motivated by intriguing questions rather than immediate engineering needs. We also consider one way to use the resulting understanding in the context of so-called ubiquitous computing.

Keywords: Process algebra, ubiquitous computing

Why need the behaviour of concurrent computing be understood? Is there any mystery? After all, however powerful they are, our programming languages and computing machines are designed by humans. So they are the result of human engineering, just as, say, cars are. Is there any mystery how your car runs? Our cars were made to run by capable engineers' hands, who may have relied on physical laws of the universe, but these laws are a pre-requisite to the engineering of cars rather than part of it, so not much of a "science of cars" seems possible. Then how can we have science of programs, science of software?

As in many other fields of science, it may be safe to say that our field did not begin out of sheer material needs. It started with curiosity of researchers, for example when pioneers of concurrency theory stumbled upon a question, say, how can one mathematically capture the behaviour of a simple concurrent program such as $x := 1; y := 2x$ in parallel with $x := 3$. It turned out that this is an astoundingly deep problem, long lasting and invoking one question after another, leading to many calculi, many new notions of relations, a new set theory, and new deduction methods. Taking an example with which I am familiar, one of the significant findings out of these inquiries is that the idea of concurrent processes which communicate with each other solely by sending and receiving communication channel names and nothing

[1] Email:kohei@dcs.qmul.ac.uk

doi:10.1016/j.entcs.2006.01.032

else, gives rise to a very expressive calculus that can precisely represent a wide range of behaviours of concurrent and sequential computation. Interestingly, this calculus turned out to be closely associated with another significant recent discovery in a related field, a solution to the so-called "full abstraction problem", which is about faithfulness of mathematical models of conventional sequential programming languages.

The science of natural numbers, Number Theory, also started in the same way, out of curiosity: the subject initially started, and still proceeds, through curiosity of inquiring minds, by stumbling upon some questions, perhaps posed by oneself, perhaps whispered by others. The richness and depth of this mathematical structure, the natural numbers, could only be appreciated after its study matured.

But computing science in general and process theory in particular are not only about mathematical structures. They are first and foremost about computation. It is the desire to fully understand how concurrent processes and, more generally, computing agents behave (and how a rich collection of behaviours they have!) that drive those in this discipline. And it is also directly about engineering. The reason why a general inquiry into behaviour of software matters in engineering is because the primary tool for software engineering, programming languages, can realise an infinite variety of (tangible and sometimes intangible) software behaviour. The resulting "design space" of behaviours, even restricted to what can be generated by a single programming language such as Pascal, Java or ML, is extremely large, so much so that even just classifying all possible realisable software behaviours in an orderly way can become an intellectual challenge, along with other questions such as specifying their properties in a meaningful way using, say, logical formulae. Simply put, in software engineering, the variety with which we can combine components, as well as the variety of available components themselves, is too large to handle without general principles. This singular nature of computing systems in general and software in particular is the reason why we need to found our engineering principles on a general mathematical basis.

Thus saying the behaviours of programs in Java, C, or ML are well-understood (after all their definitions have been written down, either as informal standards or as rigorous mathematical definitions), is the same thing as saying there is no mystery in natural numbers, for the reason that we know how to count them. That you can count (and all numbers look just so plain) is certainly true but that does not contradict that there are many inexhaustible questions on numbers remaining to be solved. In fact some basic aspects of even sequential programs are so subtle that their clarification demanded long years of study (the full abstraction problem mentioned before is one of such problems): when it comes to concurrency and communication, which encompass a far wider variety of behaviours, many aspects of this broad terrain are awaiting to be uncovered, on the basis of accumulated study of theories of processes.

I started this note with the title "In the Age of Ubiquitous Computing". In ubiquitous computing, we are surrounded by numerous and often invisible digital agents which communicate with each other and which may proactively offer service and

change environment to us. As Stajano and Crowcroft examine in their recent essay [2], such an environment-service complex poses fundamental problems about, for example, privacy (how can you guarantee privacy when you are always being seen, felt and heard by digital sensory organs of the environment?) and responsibility (if you let your car drive for itself and if it has an accident by some malfunctioning, who takes responsibility?). The infrastructure can become overwhelmingly powerful and can easily be abused, so that it can even become a basic threat to our civilised life. In the same context Stajano [1] writes:

> It would be evil if pervasive surveillance were built into ubicomp on purpose, but it would be tragically idiotic if this just happened by negligence — simply because thinking of appropriate safeguards was too hard and therefore too expensive.

Such safeguards can only be materialised by maturing our engineering and social understanding of the underlying issues, and by formulating clear and implementable engineering criteria, as well as making them understood by society at large. Good engineering ideas are certainly important, but if we cannot describe behaviour of computing agents clearly, there is no hope we can even agree on in what way the behaviour of, say, your personal electronic assistant should be (which vendors should engineer and sell following a standard), for example for you to be sure it does not violate your privacy. It is true that corporate executives can have bugs even now in their offices: but in the world where your living room will be constantly downloading components from the outside which are connected to sensory machinery and may communicate with the outside, the degree of potential privacy violation will be much greater. In fact it is not only privacy but also general safety of software behaviour which is at stake, because privacy violation is but a single manifestation of how crucially and intricately our daily lives will be relying on computing, whose key features will predominantly include communication and concurrency. Describing behaviours of sequential and concurrent software and controlling them, up to the precision all able engineers can agree on, is surely one place where science is demanded.

Science cannot survive without intriguing questions. Theories of processes, including process algebras and calculi, thrive on these questions, which get unexpectedly related with other threads of research such as semantics of sequential programs, again leading to new questions. The true life of a field of science only exists in such intellectual dynamics. Science is definitely for understanding. At the same time, it is partly because of unexpected use of such understanding for enrichment of human life (in many kinds) that society can maintain these activities.

Promoting the shared understanding of the engineering principles for building software for information appliances is far from a unique challenge to theories of processes in the context of ubiquitous computing. In fact, even this subject itself, which is more socially than scientifically oriented, poses us the same basic questions about behaviours of communicating computing agents as the theory of processes has posed in its long history, with a new twist in such elements as real-time and location. What is interactive behaviour? How do we specify its properties? When can we substitute one sub-behaviour for another without affecting the whole behaviour?

How can we compose behaviours and what is the result of composition? These are classical questions, which acquire new life in the context of the urgent social and ethical needs of ubiquitous computing.

References

[1] Stajano, F. Security for Whom? The shifting Security Assumptions of Pervasive Computing. *ISSS 2002*, LNCS 2609, pp. 16-27, Springer-Verlag, 2003.

[2] Stajano, F. and Crowcroft, J. The butt of the iceberg: Hidden Security Problems of Ubiquitous Systems. *Ambient intelligence: impact on embedded system design*, pp. 91–101, Kluwer Academic Publishers, 2003.